Semantic Data Integration in P2P Systems

Diego Calvanese, Elio Damaggio, Giuseppe De Giacomo,
Maurizio Lenzerini, Riccardo Rosati

Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113, I-00198 Roma, Italy
lastname@dis.uniroma1.it

Abstract. In this paper, we study the problem of data integration in P2P systems. Differently from the traditional setting, data integration in these systems is not based on the existence of a global view. Instead, each peer exports data in terms of its own schema, and information integration is achieved by establishing mappings among the various peer schemas. We present a framework that captures this general architecture, and then we discuss the problem of characterizing the semantics of such framework. We show that the usual approach of resorting to a first-order logic intepretation of P2P mappings, leads both to a poor modeling of the whole system, and to undecidability of query answering, even for mappings of a restricted form. This motivates the need of a new semantics for P2P system. We then present a novel proposal, based on epistemic logic, and show that not only it adequately models the interactions among peers, but it also supports decidable query answering. In particular, for the restricted form of mapping mentioned above, query answering is polynomial with respect to the size of data stored in the peers.

1 Introduction

Most of the formal approaches to data integration refer to an architecture based on a global schema and a set of sources. The sources contain the real data, while the global schema provides a reconciled, integrated, and virtual view of the underlying sources. One of the challenging issues in these systems is to answer queries posed to the global schema. Due to the architecture of the system, query processing requires a reformulation step: the query over the global schema must be re-expressed in terms of a set of queries over the sources [12, 13, 17, 15].

In this paper, we study the problem of data integration in peer-to-peer systems. In these systems every node (peer) acts as both client and server, and provides part of the overall information available from a distributed environment, without relying on a single global view. A suitable infrastructure is adopted for managing the information in the various nodes. Napster[1], which made the P2P idea popular, employs a centralized database with references to the information

[1] http://www.napster.com/

items (files) on the peers. Gnutella, another well-known P2P system, has no central database, and is based on a communication-intensive search mechanism. More recently, a Gnutella-compatible P2P system, called Gridella [1], has been proposed, which follows the so-called Peer-Grid (P-Grid) approach. A P-Grid is a virtual binary tree that distributes replication over community of peers and supports efficient search. P-Grid's search structure is completely decentralized, supports local interactions between peers, uses randomized algorithms for access and search, and ensures robustness of search against node failures.

As pointed out in [10], current P2P systems focus strictly on handling semantic-free, large-granularity requests for objects by identifier, which both limits their utility and restricts the techniques that might be employed to distribute the data. These current sharing systems are largely limited to applications in which objects are described by their name, and exhibit strong limitations in establishing complex links between peers. To overcome these limitations, data-oriented approaches to P2P have been proposed recently [11, 2, 10]. For example, in the Piazza system [10], data origins serve original content, peer nodes cooperate to store materialized views and answer queries, nodes are connected by bandwidth-constrained links and advertise their materialized views to share resources with other peers.

Differently from the traditional setting, integration in data-oriented P2P systems is not based on a global schema. Instead, each peer represents an autonomous information system, and information integration is achieved by establishing P2P mappings, i.e., mappings among the various peers. Queries are posed to one peer, and the role of query processing is to exploit both the data that are internal to the peer, and the mappings with other peers in the system. To stress the data-oriented nature of the framework, we refer to "semantic data integration", in the sense that we assume that the various peers export data in terms of a suitable schema, and mappings are established among such peer schemas. A peer schema is therefore intended to export the semantics of information as viewed from the peer.

In this paper, we present a framework that captures this general architecture (Section 2), and then discuss the problem of characterizing the semantics of such framework (Section 3). We argue that, although correct from a formal point of view, the usual approach of resorting to a first-order logic interpretation of P2P mappings (followed for example by [6, 11, 2]), has several drawbacks, both from the modeling and from the computational perspective. In particular, query answering under the first-order semantics is undecidable, even for restricted forms of P2P systems, called *simple* P2P systems, i.e., for the case where the various peers are empty first-order theories. Motivated by these observations, several authors proposed suitable limitations to the form of P2P mappings, such as acyclicity.

Based on the above mentioned drawbacks, we propose a new semantics for P2P systems, with the following aims: *(i)* we want to take into account that peers are autonomous modules, and the modular structure of the P2P system should be explicitly reflected in the definition of its semantics; *(ii)* we do not

want to limit a-priori the topology of the mapping assertions between the peers in the system; *(iii)* we seek for a semantic characterization that leads to a setting where query answering is decidable, and possibly, polynomially tractable.

We base our proposal of a new semantics for P2P systems on epistemic logic, and we show that not only it adequately models the interactions among peers, but it also supports decidable query answering. In particular, for simple P2P systems, we devise a query answering algorithm which runs in polynomial time with respect to the size of data stored in the peers (Section 4).

2 Framework

In this section, we set up the general framework for peer-to-peer (P2P) systems. We refer to a fixed, infinite, denumerable, set Γ of constants. Such constants are shared by all peers, and are the constants that can appear in the P2P system. Moreover, given a (relational) alphabet A, we denote with $\mathcal{L}_A$ the set of function-free first-order logic (FOL) formulas whose relation symbols are in A and whose constants are in Γ. A *FOL query* of arity n over an alphabet A is written in the form

$$\{\mathbf{x} \mid \mathit{body}(\mathbf{x})\}$$

where $\mathit{body}(\mathbf{x})$ is an open formula of $\mathcal{L}_A$ with free variables $\mathbf{x} = x_1, \ldots, x_n$. A *conjunctive query* (CQ) of arity n over an alphabet A is written in the form

$$\{\mathbf{x} \mid \exists \mathbf{y}\ \mathit{cbody}(\mathbf{x}, \mathbf{y})\}$$

where $\mathit{cbody}(\mathbf{x}, \mathbf{y})$ is an conjunction of atoms of $\mathcal{L}_A$ involving the free variables (also called the distinguished variables of the query) $\mathbf{x} = x_1, \ldots, x_n$, the existentially quantified variables (also called the non-distinguished variables of the query) $\mathbf{y} = y_1, \ldots, y_m$, and constants from Γ.

A P2P system is constituted by a set of peers, and a set of mappings among peers. We first concentrate on describing the structure of a single peer. Following the basic ideas presented in [11], we define a peer P as a triple $P = (G, S, L)$, where:

- G is the *schema* of P, which is defined, starting from an alphabet A_G, as a set of formulas of $\mathcal{L}_{A_G}$. We call A_G the *alphabet* of P.
- S is the *(local) source schema* of P, that is simply a finite relational alphabet, which is called the *local alphabet* of P.
- L is a set of *(local) mapping assertions* between G and S. Each local mapping assertion has the form

  $$cq_S \rightsquigarrow cq_G$$

  where cq_S and cq_G are two conjunctive queries of the same arity, over the source schema S and over the peer schema G, respectively.

Intuitively, the source schema describes the structure of the data sources of the peer (possibly obtained by wrapping physical sources), where the real data

managed by the peer are stored, while the peer schema provides a virtual view of the information managed by, and exported by the peer. The mapping assertions establish the connection between the elements of the source schema and those of the peer schema. In particular, an assertion of the form $cq_S \rightsquigarrow cq_G$ specifies that all the data satisfying the query cq_S over the sources also satisfy the concept in the peer schema represented by the query cq_G. This form of mapping is the most expressive one among those studied in the data integration literature. Indeed, in terms of the terminology used in data integration, a peer in our setting corresponds to a GLAV *data integration system* [9] managing a set of sound data sources S defined in terms of a (virtual) global schema G.

A P2P system is constituted by a set of peers and a set of mappings that specify the semantic relationships between the data exported by the peers. Formally, a *P2P system* is a pair $\mathcal{S} = (\mathcal{P}, \mathcal{M})$, where $\mathcal{P}$ is a finite set of peers, and $\mathcal{M}$ is a finite set of P2P-mapping assertions. Each *P2P mapping assertion* has the form

$$q_1 \rightsquigarrow q_2$$

where

- q_1, called the *tail* of the assertion, is a FOL query over the union of the alphabets of the peers in $\mathcal{P}$,
- q_2, called the *head* of the assertion, is a FOL query over the alphabet of a single peer, and
- q_1 and q_2 are of the same arity.

Intuitively, a P2P mapping assertion $q_1 \rightsquigarrow q_2$, where q_2 is a query over the schema of a peer P, expresses the fact that P can use, besides the data in its local sources, also the data retrieved by q_1 from the peers over which q_1 is expressed. Such data are mapped to the schema of P according to what is specified by the query q_2. Observe that P2P mapping assertions may be cyclic, in the sense that no limitation is imposed on the graph representing such assertions. This graph contains one node for every relation symbol in the peer schemas, and one arc from the node corresponding to R_1 to the node corresponding to R_2 if there is a P2P mapping assertion whose tail mentions R_1 and whose head mentions R_2.

Finally, we assume that queries are posed to a single peer of the system. More precisely, a *query* over a P2P system $\mathcal{S} = (\mathcal{P}, \mathcal{M})$ is a first-order query over the peer schema of a single peer in $\mathcal{P}$.

3 Semantics

In this section, we first define the semantics of one peer (subsection 3.1), and then we discuss two mechanisms for specifying the semantics of the whole P2P system. The first mechanism (subsection 3.2), adopted in most formal approaches to P2P data integration, is based of FOL. Motivated by several drawbacks of this approach, we propose a new semantics, based on epistemic logic (subsection 3.3).

In what follows, a database (DB) for a schema $\mathcal{T}$ is simply a set of collection of relations, one for each symbol in the alphabet of $\mathcal{T}$. Also, if q is a query of

arity n and $\mathcal{DB}$ is a database, we denote with $ans(q, \mathcal{DB})$ the set of tuples (of arity n) in $\mathcal{DB}$ that satisfy q.

3.1 Semantics of one peer

In order to assign formal meaning to a peer $P = (G, S, L)$, we conceive P as a FOL theory T_P, called "peer theory", defined as follows:

- The alphabet of T_P is obtained as union of the alphabet $\mathcal{A}_\mathcal{G}$ of G and the alphabet of the local sources S of P,
- The formulas of T_P are obtained as follows:
  - there is one formula of the form

$$\forall \mathbf{x}\ (cq_S(\mathbf{x}) \rightarrow cq_G(\mathbf{x}))$$

    for each local mapping assertion $cq_S \rightsquigarrow cq_G$ in L
  - T_P includes all the FOL formulas expressing the schema G.

We make a simplifying assumption on the domain of the various databases. In particular, we assume that the databases involved in our framework (both the databases conforming to the local schemas, and those conforming to the peer schemas) share the same infinite domain Δ, fixed once and for all. We also assume that the constants in Γ (see the previous section) have the same, fixed, interpretation in all databases, i.e., to each constant $c \in \Gamma$ is associated, once and for all, a certain domain element $d \in \Delta$. Moreover, we assume that Γ contains a constant for each element in Δ, and that different constants are interpreted as different domain elements. It follows that Γ is actually isomorphic to Δ, so that we can use (with a little abuse of notation) constants in Γ whenever we want to denote domain elements.[2]

Now, the semantics of P directly follows from its characterization in FOL, and the assumption above on the interpretation domain. However, in order to point out the role of local sources in P, we specialize the notion of FOL semantics by starting with a *local source database* for P, i.e., a finite database $\mathcal{D}$ for the source schema S. Based on $\mathcal{D}$, we now specify which is the information content of the peer schema G at the extensional level. We call *peer database* for P any database for G. A peer database $\mathcal{B}$ for P is said to be a *model of P with respect to $\mathcal{D}$* if:

- $\mathcal{B}$ satisfies all the formulas expressing the meaning of G,
- $\mathcal{B}$ satisfies every mapping assertion in L, where $\mathcal{B}$ satisfies the mapping assertion $cq_S \rightsquigarrow cq_G$ if every tuple that satisfies cq_S in $\mathcal{D}$ satisfies also cq_G in $\mathcal{B}$.

Finally, we specify the semantics of queries posed to a peer. As we said before, such queries are expressed in terms of the alphabet $\mathcal{A}_\mathcal{G}$, i.e., in terms of the symbols in the peer schema G of P. Given a local source database $\mathcal{D}$ for P,

[2] In other words the constants in Γ act as *standard names* [16].

the answer $ans(q, P, \mathcal{D})$ to a query q in P with respect to $\mathcal{D}$, is the set of tuples $\mathbf{t}$ of constants in Γ such that $t \in ans(q, \mathcal{B})$ for *every* peer database $\mathcal{B}$ that is a model of P with respect to $\mathcal{D}$. The set $ans(q, P, \mathcal{D})$ is called the set of *certain answers* to q in P with respect to $\mathcal{D}$.

In the next two subsections, we turn our attention to the problem of specifying the semantics of the whole P2P system.

3.2 FOL semantics for P2P systems

The first approach we discuss for assigning semantics to a P2P system, is the FOL approach, followed by [6, 14, 11]. In this approach, one associates to a P2P system $\mathcal{S} = (\mathcal{P}, \mathcal{M})$ a *single* FOL theory, obtained as the union of the various peer theories, plus suitable FOL formulas corresponding to the P2P mapping assertions. In particular, we have one mapping formula

$$\forall \mathbf{x} \;\; (q_1(\mathbf{x}) \rightarrow q_2(\mathbf{x}))$$

for each P2P mapping assertion $q_1 \rightsquigarrow q_2$.

In this formalization, the models of the whole P2P system $\mathcal{S}$ are simply the FOL models of the corresponding FOL theory. Although correct from a formal point of view, we argue that this formalization has two main drawbacks:

- Since this approach considers the whole P2P system as a single flat FOL theory, in the formalization the structure of the system in terms of peers is actually lost. In other words, the formulas of the various peers, and the mapping between peers become formulas of the theory, without any formal distinction of their roles.
- One of the implications of the above observation is that, even if we restrict the general framework in such a way that the various peers are *decidable* FOL theories (in particular, empty theories), query answering in the whole P2P system is actually undecidable, as illustrated in the next section. This is why, in [11, 14], the authors propose syntactic restrictions on the mapping assertions (e.g., acyclicity).

The above drawbacks suggest that it is worth exploring other kinds of semantics for P2P systems.

3.3 A new semantics for P2P systems based on epistemic logic

Based on the above mentioned drawbacks, we propose a new semantics for P2P systems, with the following aims:

- We want to take into account that peers in our context are to be considered autonomous sites, that exchange information. In other words, peers are modules, and the modular structure of the system should be explicitly reflected in the definition of its semantics.

– We do not want to limit a-priori the topology of the mapping assertions among the peers in the system. In particular, we do not want to impose acyclicity of assertions.
– We seek for a semantic characterization that leads to a setting where query answering is decidable, and possibly, polynomially tractable.

We base our proposal of a new semantics for P2P systems on epistemic logic[3]. Due to space limitations, we cannot delve into the details of epistemic logic here. We simply describe its basic notions. In epistemic logic, the language is the one of FOL, except that, besides the usual atoms, one can use another form of atoms, namely:

$$\mathbf{K}\alpha$$

where α is again a formula. Intuitively, the formula $\mathbf{K}\alpha$ is interpreted as the objects that are *known* to satisfy α, i.e., that satisfy α in all possible FOL models (of the kind seen so far, in our case).

Formally, the semantics of an epistemic logic theory is based on the notion of epistemic interpretation. We remind the reader that we are referring to a unique interpretation domain Γ. An epistemic interpretation $\mathcal{E}$ is a pair $(\mathcal{I}, \mathcal{W})$, where $\mathcal{I}$ is a FOL interpretation, $\mathcal{W}$ is a set of FOL interpretations, and $\mathcal{I} \in \mathcal{W}$. The notion of satisfaction of a formula in an epistemic interpretation $\mathcal{E} = (\mathcal{I}, \mathcal{W})$ is analogous to the one in FOL, with the provision that the interpretation for the atoms is as follows:

– a FOL formula constituted by an atom $f(\mathbf{x})$ (where $\mathbf{x}$ are the free variables in F) is satisfied in $(\mathcal{I}, \mathcal{W})$ by the tuples of constants $\mathbf{t}$ such that $f(\mathbf{t})$ is true in $\mathcal{I}$,
– an atom of the form $\mathbf{K}\alpha(\mathbf{x})$ is satisfied in $(\mathcal{I}, \mathcal{W})$ by the tuples of constants $\mathbf{t}$ such that $\alpha(\mathbf{t})$ is satisfied in all the pairs $(\mathcal{J}, \mathcal{W})$ such that $\mathcal{J} \in \mathcal{W}$.

An *epistemic model* of an epistemic logic theory is an epistemic interpretation that satisfies every formula of the theory.

Observe that in epistemic logic the formula $\mathbf{K}(\alpha \vee \beta)$ has an entirely different meaning with respect to the formula $\mathbf{K}\alpha \vee \mathbf{K}\beta$. Indeed, the former is satisfied in an interpretation $(\mathcal{J}, \mathcal{W})$ if for every $\mathcal{I} \in \mathcal{W}$, there is at least one among $\{\alpha, \beta\}$, α or β is satisfied in $\mathcal{I}$. Conversely, the latter requires that there is one formula among α and β that is satisfied in all $\mathcal{I} \in W$. Observe also that, if α is a FOL formula, there is a striking difference between $\mathbf{K}\exists x.\alpha(x)$ and $\exists x.\mathbf{K}\alpha(x)$. In particular, for $\exists x.\mathbf{K}\alpha(x)$ to be satisfied in $(\mathcal{I}, \mathcal{W})$ there must be a constant $c \in \Gamma$ such that $\alpha(c)$ is satisfied in every $J \in \mathcal{W}$.

We formalize a P2P system $\mathcal{S} = (\mathcal{P}, \mathcal{M})$ in terms of the epistemic logic theory $E_{\mathcal{S}}$, called P2P theory, constructed as follows:

– the alphabet is the disjoint union of the alphabets of the various peer theories, one corresponding to one peer in $\mathcal{P}$[4]

[3] Technically we resort to epistemic FOL with standard names, and therefore with a fixed domain, and rigid interpretation of constants [16].

[4] In order to get the disjoint union, we may simply rename the predicates of the schemas of the various peers by prefixing the peer identifier.

– all the formulas of the various theories T_P belong to $E_{\mathcal{S}}$,
– there is one formula in $E_{\mathcal{S}}$ of the form

$$\forall \mathbf{x} \;\; ((\mathbf{K}\, q_1(\mathbf{x})) \rightarrow q_2(\mathbf{x}))$$

for each P2P mapping assertion $q_1 \rightsquigarrow q_2$ in $\mathcal{M}$.

Note that the formalization of the P2P mapping assertions in terms of the formulas specified above intuitively reflects the idea that only what is *known* by the peers mentioned in the tail of the assertion is transferred to the peer mentioned in the head.

Now, the semantics of the P2P system $\mathcal{S}$ directly follows from the above characterization in epistemic logic. However, as we did for the case of one peer, in order to point out the role of local sources in the various peers, we specialize the notion of epistemic semantics by starting with a collection of source databases, one for each peer in $\mathcal{P}$

Let $\mathcal{D}_1, \ldots, \mathcal{D}_n$ be n local source databases for the peers $P_1, \ldots, P_n$ in $\mathcal{P}$. We call *source database* $\mathcal{D}$ *for* $\mathcal{S}$ based on $\mathcal{D}_1, \ldots, \mathcal{D}_n$ the disjoint union of $\mathcal{D}_1, \ldots, \mathcal{D}_n$. Moreover, let $\mathcal{B}_1, \ldots, \mathcal{B}_n$ be n models of $P_1, \ldots, P_n$ with respect to $\mathcal{D}_1, \ldots, \mathcal{D}_n$, respectively. The disjoint union of $\mathcal{B}_1, \ldots, \mathcal{B}_n$ is called a *FOL model* for $\mathcal{S}$ based on $\mathcal{D}$.

We can now introduce the notion of epistemic interpretation for $\mathcal{S}$: an *epistemic interpretation* for $\mathcal{S}$ based on $\mathcal{D}$ is a pair $(\mathcal{I}, \mathcal{W})$ such that $\mathcal{I}$ is a FOL model for $\mathcal{S}$ based on $\mathcal{D}$, $\mathcal{W}$ is a set of FOL models for $\mathcal{S}$ based on $\mathcal{D}$, and $\mathcal{I} \in \mathcal{W}$. Taking into account the semantics of epistemic logic described above, it is easy to see that an *epistemic model* for $\mathcal{S}$ based on $\mathcal{D}$ is any epistemic interpretation for $\mathcal{S}$ based on $\mathcal{D}$ that satisfies all the epistemic formulas corresponding to the P2P mapping assertions in $\mathcal{M}$. In particular, an epistemic interpretation $(\mathcal{I}, \mathcal{W})$ for $\mathcal{S}$ based on $\mathcal{D}$ satisfies the P2P mapping assertion $q_1 \rightsquigarrow q_2$ if, for every tuple $\mathbf{t}$ of objects in Γ, the fact that $q_1(\mathbf{t})$ is satisfied in every FOL models in $\mathcal{W}$ implies that $q_2(\mathbf{t})$ is satisfied in $\mathcal{I}$.

Let q be a query over one peer of $\mathcal{S}$. The certain answer $ans_{\mathbf{k}}(q, \mathcal{S}, \mathcal{D})$ to q in $\mathcal{S}$ based on $\mathcal{D}$ is the set of tuples $\mathbf{t}$ of objects in Γ such that $q(\mathbf{t})$ is satisfied in every epistemic model $(\mathcal{I}, \mathcal{W})$ of $\mathcal{S}$ based on $\mathcal{D}$, i.e., the set of tuples $\mathbf{t}$ of objects in Γ such that, for every every epistemic model $(\mathcal{I}, \mathcal{W})$ of $\mathcal{S}$ based on $\mathcal{D}$, $q(\mathbf{t})$ is satisfied in $\mathcal{I}$.

4 Query Answering

In this section we address query answering in P2P systems. We start by noticing that query answering in the general framework is obviously undecidable. Indeed, since peer schemas are arbitrary FOL theories, it is undecidable even to answer boolean queries posed to a P2P system constituted by a single peer. So, it makes sense to introduce some restriction to make query answering more manageable.

Here we focus on a specific restricted setting. In particular, we call a P2P system $\mathcal{S} = (\mathcal{P}, \mathcal{M})$ *simple* if it satisfies the following restrictions:

1. peer theories are empty, i.e., each peer schema of $\mathcal{S}$ simply consists of a relational alphabet;
2. P2P mapping assertions in $\mathcal{M}$ are expressed using conjunctive queries, i.e., a P2P mapping assertion is an expression of the form $q_1 \rightsquigarrow q_2$, where q_1 and q_2 are conjunctive queries of the same arity, q_1 is expressed over the union of the alphabets of the peers, and q_2 is expressed over the alphabet of a single peer.
3. the language for querying the P2P system is *union of conjunctive queries (UCQ)*, i.e., a query over a P2P system is a UCQ over the alphabet of a single peer.

Such a restricted framework allows us to isolate the complexity coming from the P2P mappings. Indeed, since we have dropped constraints (axioms) in the various peer schemas, we are avoiding the introduction of complexity coming from the structure of such schemas. Also, having restricted the queries used in the P2P mapping and the queries posed to the P2P system to conjunctive (union of conjunctive) queries, which are well investigated in data integration, allows us to understand the complexity coming out of the core structure of the P2P system itself.

Interestingly, if we adopt the FOL semantics for simple P2P systems, query answering remains undecidable. This is mainly due to the presence of cycles in the P2P mapping assertions [7].

One of the contributions of our work is that, if we instead adopt the epistemic semantics, we get decidability for query answering. We show this by providing a sound and complete algorithm for query answering in simple P2P systems.

In particular, we show that, given an UCQ q posed to a simple P2P system S, and given a source database $\mathcal{D}$ for $\mathcal{S}$, one can construct a finite (relational) database RDB on the alphabet $\mathcal{A}_\mathcal{S}$ that is the union of the alphabet of the peer schemas in $\mathcal{S}$, such that for each tuple $\mathbf{t}$ of constants in Γ, $\mathbf{t} \in ans_\mathbf{k}(q, \mathcal{S}, \mathcal{D})$ if and only if $\mathbf{t} \in ans(q, RDB)$. Intuitively, such a finite database RDB constitutes a "representative" of all the epistemic models for $\mathcal{S}$ based on $\mathcal{D}$ with respect to the query q.

The database RDB contains, as objects, constants in Γ and new constants, called below *fresh values*, coming from an infinite, denumerable set of constants Φ, disjoint from Γ. To construct RDB, we make use of the following algorithm:

Algorithm build-rdb$(\mathcal{S}, \mathcal{D})$
Input: simple P2P system $\mathcal{S} = (\mathcal{P}, \mathcal{M})$, with $\mathcal{P} = \{P_1, \ldots, P_n\}$,
source database $\mathcal{D}$ for $\mathcal{S}$
Output: database RDB on $\mathcal{A}_\mathcal{S}$
$RDB \leftarrow \emptyset$;
(a) **for** $i = 1, \ldots, n$ **do** $RDB \leftarrow$ retrieve-local-mapping$(\mathcal{S}, P_i, D, RDB)$;
(b) **repeat**
$RDB' \leftarrow RDB$;
$RDB \leftarrow$ retrieve-P2P-mapping$(\mathcal{M}, RDB)$
until $RDB' = RDB$;
return RDB

Informally, the algorithm proceeds as follows: first, through the local mapping assertions, it retrieves data from the local sources of the peers, and stores such data in the database *RDB*; then, through the P2P mapping assertions, it adds new data to *RDB*, until no new data can be added.

To compute the database *RDB*, the algorithm resorts to two subroutines, retrieve-local-mapping and retrieve-P2P-mapping. The first one retrieves data from the local sources of the peers according to the local mapping assertions, while the second one derives new data according to the P2P mapping assertions. In order to define such operations, we introduce the following notation. Given a conjunctive query $q = \{\mathbf{x} \mid \exists \mathbf{y}.\mathit{cbody}(\mathbf{x}, \mathbf{y})\}$, a tuple $\mathbf{t}$ of the same arity of $\mathbf{x}$, and a database $\mathcal{DB}$ whose objects are in $\Gamma \cup \Phi$, we denote with $\mathit{fresh}(q, \mathbf{t}, \mathcal{DB})$ the set of atoms obtained by instantiating the distinguished variables in q by $\mathbf{t}$ and the existentially quantified variables in q by some fresh values not already occurring in $\mathcal{DB}$. In other words, $\mathit{fresh}(q, \mathbf{t}, \mathcal{DB})$ is the set of atoms that form the conjuncts of $\mathit{cbody}(\mathbf{t}, \mathbf{v})$ where each v in $\mathbf{v}$ is a fresh value not occurring in $\mathcal{DB}$.

Below we define the algorithm retrieve-local-mapping that, given a simple P2P system $\mathcal{S} = (\mathcal{P}, \mathcal{M})$, a peer $P = (G, S, L)$ in $\mathcal{P}$, a source database $\mathcal{D}$ for $\mathcal{S}$ and a database *RDB* on $\mathcal{A}_\mathcal{S}$, adds to *RDB* all the facts that are consequence of the local mapping assertions in L evaluated over $\mathcal{D}$.

Algorithm retrieve-local-mapping($\mathcal{S}, P, \mathcal{D}, \mathit{RDB}$)
Input: simple P2P system $\mathcal{S} = (\mathcal{P}, \mathcal{M})$,
peer $P = (G, S, L)$ in $\mathcal{P}$,
source database $\mathcal{D}$ for $\mathcal{S}$,
database *RDB* on $\mathcal{A}_\mathcal{S}$
Output: database *RDB'* on $\mathcal{A}_\mathcal{S}$
RDB' $\leftarrow$ *RDB*;
for each $q_s \rightsquigarrow q_g \in L$ **do**
  for each $\mathbf{t} \in \mathit{ans}(q_s, \mathcal{D})$ **do**
    if $\mathbf{t} \notin \mathit{ans}(q_g, \mathit{RDB'})$
    then *RDB'* $\leftarrow$ *RDB'* $\cup\, \mathit{fresh}(q_g, \mathbf{t}, \mathit{RDB'})$
return *RDB'*

Then, we define the algorithm retrieve-P2P-mapping that, given a set of P2P mapping assertions $\mathcal{M}$ and a database *RDB* on $\mathcal{A}_\mathcal{S}$, adds to *RDB* all the facts that are consequences of the assertions $\mathcal{M}$ evaluated over *RDB*.

Algorithm retrieve-P2P-mapping($\mathcal{M}, \mathit{RDB}$)
Input: P2P mapping assertions $\mathcal{M}$,
database *RDB* on $\mathcal{A}_\mathcal{S}$
Output: database *RDB'* on $\mathcal{A}_\mathcal{S}$
RDB' $\leftarrow$ *RDB*;
for each $q \rightsquigarrow q_i \in \mathcal{M}$ **do**
  for each tuple $\mathbf{t}$ of constants in Γ such that $\mathbf{t} \in \mathit{ans}(q, \mathit{RDB'})$ **do**
    if $\mathbf{t} \notin \mathit{ans}(q_i, \mathit{RDB'})$
    then *RDB'* $\leftarrow$ *RDB'* $\cup\, \mathit{fresh}(q_i, \mathbf{t}, \mathit{RDB'})$
return *RDB'*

The next theorem proves termination of the algorithm build-rdb.

Theorem 1. *Let $\mathcal{S} = (\mathcal{P}, \mathcal{M})$ be a simple P2P system, $\mathcal{D}$ a source database for $\mathcal{S}$, and q a UCQ of arity n over the alphabet of a single peer in $\mathcal{P}$. Then, the algorithm build-rdb$(\mathcal{S}, \mathcal{D}, q)$ terminates, and returns a finite database RDB on $\mathcal{A}_\mathcal{S}$ whose objects are constants in Γ and fresh values in Φ.*

The next theorem gives us soundness and completeness of the technique presented here, with respect to the epistemic semantics, for simple P2P systems.

Theorem 2. *Let $\mathcal{S} = (\mathcal{P}, \mathcal{M})$ be a simple P2P system, $\mathcal{D}$ a source database for $\mathcal{S}$, q a UCQ of arity n over the alphabet of a single peer in $\mathcal{P}$, and RDB the database returned by build-rdb$(\mathcal{S}, \mathcal{D}, q)$. Then, for each tuple $\mathbf{t}$ of constants in Γ, $\mathbf{t} \in ans(q, RDB)$ if and only if $\mathbf{t} \in ans_\mathbf{k}(q, \mathcal{S}, \mathcal{D})$.*

Informally, the proof of the above theorem is based on the fact that the finite database *RDB* computed by the algorithm build-rdb constitutes a "representative" of all the (generally infinite) models for $\mathcal{S}$ and $\mathcal{D}$. Based on such a property, the evaluation of the query q with respect to $\mathcal{S}$ and $\mathcal{D}$ is equivalent to the evaluation of q over the database *RDB*.

Finally, we analyze the complexity of the algorithm with respect to the size of data stored in the peers of $\mathcal{S}$, i.e., the size of the source database $\mathcal{D}$ for $\mathcal{S}$ (data complexity). The next theorem gives us a polynomial time bound in data complexity.

Theorem 3. *Let $\mathcal{S} = (\mathcal{P}, \mathcal{M})$ be a simple P2P system, $\mathcal{D}$ be a source database for $\mathcal{S}$, q a UCQ of arity n over the alphabet of a single peer in $\mathcal{P}$, and $\mathbf{t}$ a tuple of arity n of constants in Γ. The problem of establishing whether $\mathbf{t} \in ans_\mathbf{k}(q, \mathcal{S}, \mathcal{D})$ is in PTIME in data complexity.*

The result follows from the following observations:

1. As it is immediate to verify, the algorithm retrieve-local-mapping runs in polynomial time in data complexity, and therefore the data complexity of step (a) of the algorithm build-rdb is polynomial as well.
2. In the algorithm retrieve-P2P-mapping, for each assertion $q \rightsquigarrow q_i \in \mathcal{M}$, only the answers to the query q that are tuples of constants in Γ are considered, which implies that the number of instances of the P2P mapping assertion $q \rightsquigarrow q_i$ that cause the addition of new tuples in *RDB'* is bound by c^h, where c is the number of constants occurring in the P2P source database $\mathcal{D}$ and h is the arity of the query q. Consequently, the maximum number of iterations that can be executed by step (b) of the algorithm build-rdb is bound by $a \cdot c^k$, where a is the number of assertions in $\mathcal{M}$ and k is the maximum arity of the conjunctive queries occurring in $\mathcal{M}$. Hence, step (b) of the algorithm build-rdb runs in polynomial time data complexity, and the size of the database *RDB* computed by the algorithm is also polynomial with respect to the size of the source database $\mathcal{D}$ for $\mathcal{S}$.
3. Checking whether $\mathbf{t} \in ans(q, RDB)$ is polynomial (actually LOGSPACE) in data complexity, being q an UCQ.

5 Conclusions

In this paper we have presented a general framework for data-oriented P2P systems, and we have discussed possible methods for specifying the semantics of such systems. Motivated by several drawbacks in the usual FOL formalization of data-oriented P2P systems, we have proposed a novel semantics for data integration in these systems, and we have shown that, at least for simple P2P systems, query answering under the new semantics is not only decidable, but can be done in polynomial time with respect to the size of data stored in the peers. The main objective of our work was to study the fundamental aspects of P2P data integration, and, therefore, we made several assumptions that may be too restrictive in real applications. One direction to continue our research work is to relax some of these assumptions. In particular:

- The purpose of the algorithm presented in Section 4 was to show relevant formal properties of the proposed semantics, in particular that it supports polynomial time data complexity in computing the answers to queries posed to simple P2P systems. However, the algorithm is based on a bottom-up computation that does not exploit in any way the structure of the query. While such an approach is appropriate in a data-exchange context [7, 8], where one aims at materializing the integrated data independently of a particular query, by exploiting the structure of the query one could substantially improve efficiency of query answering. Indeed, we are working on a top-down algorithm for query answering under the epistemic semantics, that is driven by the query and the structure of mappings and avoids computing integrated data that are not relevant for the query.
- Although we have assumed here that P2P mappings mention only one peer in their heads, our results can be extended to the case of more expressive forms of mappings, in particular allowing conjunctive queries over more than one peer in the head.
- Peer schemas in simple P2P systems are specified just in terms of an alphabet. Obviously, more expressive forms of schema may be needed in real settings. Interestingly, by exploiting the results presented in [3–5], it is possible to show that both the technique described in Section 4, and the query-driven algorithm mentioned above, can be generalized to a setting where peer schemas contain important classes of constraints, such as key and foreign key constraints.
- In our formal framework we assumed the existence of a single, common set of constants for denoting the interpretation domain of all the peers. In real applications, this is a too strong assumption, as the various peers are obviously autonomous in choosing the mechanisms for denoting the domain elements. The issue of different vocabularies of constants in different peers is addressed, for example, in [2], and we believe that our approach can be extended in order to incorporate such kinds of techniques to deal with this problem.
- Finally, in our current formalization, if the information that one peer provides to another peer is inconsistent with the information known by the

latter, the whole P2P system is logically inconsistent. Again, this is a strong limitation when one wants to use the framework in real applications. Data reconciliation and cleaning techniques may mitigate such a problem in some cases. More generally, to deal with this problem, we are investigating suitable extensions of the epistemic semantics presented here, in the line of [5].

References

1. K. Aberer, M. Punceva, M. Hauswirth, and R. Schmidt. Improving data access in P2P systems. *IEEE Internet Computing*, 2002.
2. P. A. Bernstein, F. Giunchiglia, A. Kementsietsidis, J. Mylopoulos, L. Serafini, and I. Zaihrayeu. Data management for peer-to-peer computing: A vision. In *Proc. of the 5th Int. Workshop on the Web and Databases (WebDB 2002)*, 2002.
3. A. Calì, D. Calvanese, G. De Giacomo, and M. Lenzerini. Data integration under integrity constraints. *Information Systems*, 2003. To appear.
4. A. Calì, D. Lembo, and R. Rosati. On the decidability and complexity of query answering over inconsistent and incomplete databases. In *Proc. of the 22nd ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2003)*, pages 260–271, 2003.
5. A. Calì, D. Lembo, and R. Rosati. Query rewriting and answering under constraints in data integration systems. In *Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI 2003)*, 2003. To appear.
6. T. Catarci and M. Lenzerini. Representing and using interschema knowledge in cooperative information systems. *J. of Intelligent and Cooperative Information Systems*, 2(4):375–398, 1993.
7. R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: Semantics and query answering. In *Proc. of the 9th Int. Conf. on Database Theory (ICDT 2003)*, pages 207–224, 2003.
8. R. Fagin, P. G. Kolaitis, and L. Popa. Data exchange: Getting to the core. In *Proc. of the 22nd ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2003)*, pages 90–101, 2003.
9. M. Friedman, A. Levy, and T. Millstein. Navigational plans for data integration. In *Proc. of the 16th Nat. Conf. on Artificial Intelligence (AAAI'99)*, pages 67–73. AAAI Press/The MIT Press, 1999.
10. S. Gribble, A. Halevy, Z. Ives, M. Rodrig, and D. Suciu. What can databases do for peer-to-peer? In *Proc. of the 4th Int. Workshop on the Web and Databases (WebDB 2001)*, 2001.
11. A. Halevy, Z. Ives, D. Suciu, and I. Tatarinov. Schema mediation in peer data management systems. In *Proc. of the 19th IEEE Int. Conf. on Data Engineering (ICDE 2003)*, 2003.
12. A. Y. Halevy. Answering queries using views: A survey. *Very Large Database J.*, 10(4):270–294, 2001.
13. R. Hull. Managing semantic heterogeneity in databases: A theoretical perspective. In *Proc. of the 16th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'97)*, pages 51–61, 1997.
14. C. Koch. Query rewriting with symmetric constraints. In *Proc. of the 2nd Int. Symp. on Foundations of Information and Knowledge Systems (FoIKS 2002)*, volume 2284 of *Lecture Notes in Computer Science*, pages 130–147. Springer, 2002.

15. M. Lenzerini. Data integration: A theoretical perspective. In *Proc. of the 21st ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2002)*, pages 233–246, 2002.
16. H. J. Levesque and G. Lakemeyer. *The Logic of Knowledge Bases*. The MIT Press, 2001.
17. J. D. Ullman. Information integration using logical views. In *Proc. of the 6th Int. Conf. on Database Theory (ICDT'97)*, volume 1186 of *Lecture Notes in Computer Science*, pages 19–40. Springer, 1997.